Discord Bot

RAG-Based Question Answering System

Jibin Kunjumon

Problem & Objective

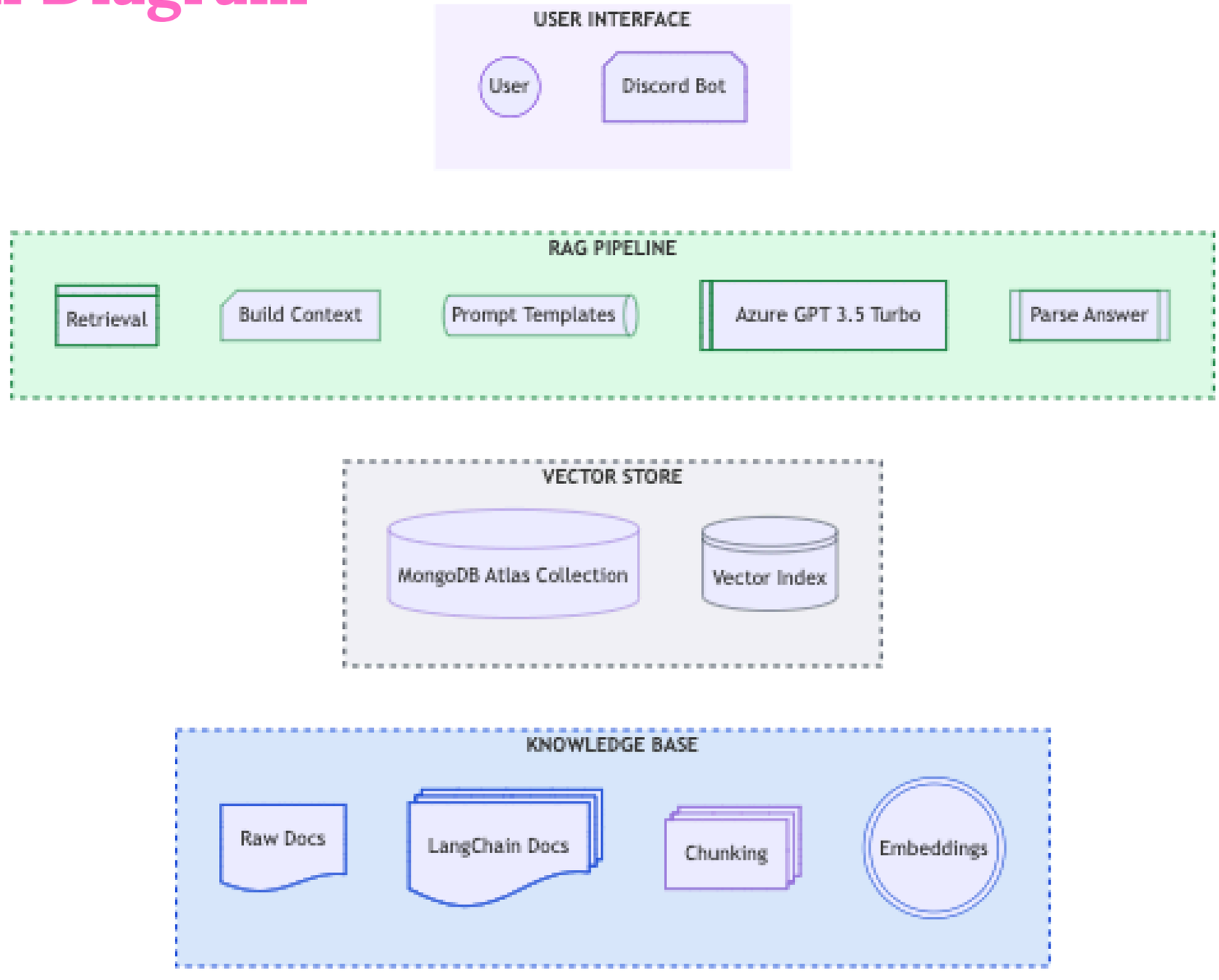
🎯 What It Does

An intelligent Discord bot that answers user questions using Retrieval-Augmented Generation (RAG) with real-time context understanding.

💡 Why It Matters

Provides instant, accurate responses by combining document retrieval with AI language models, enabling seamless knowledge access through a familiar chat interface.

Architectural Diagram



Workflow Diagram

📁 Data Preparation

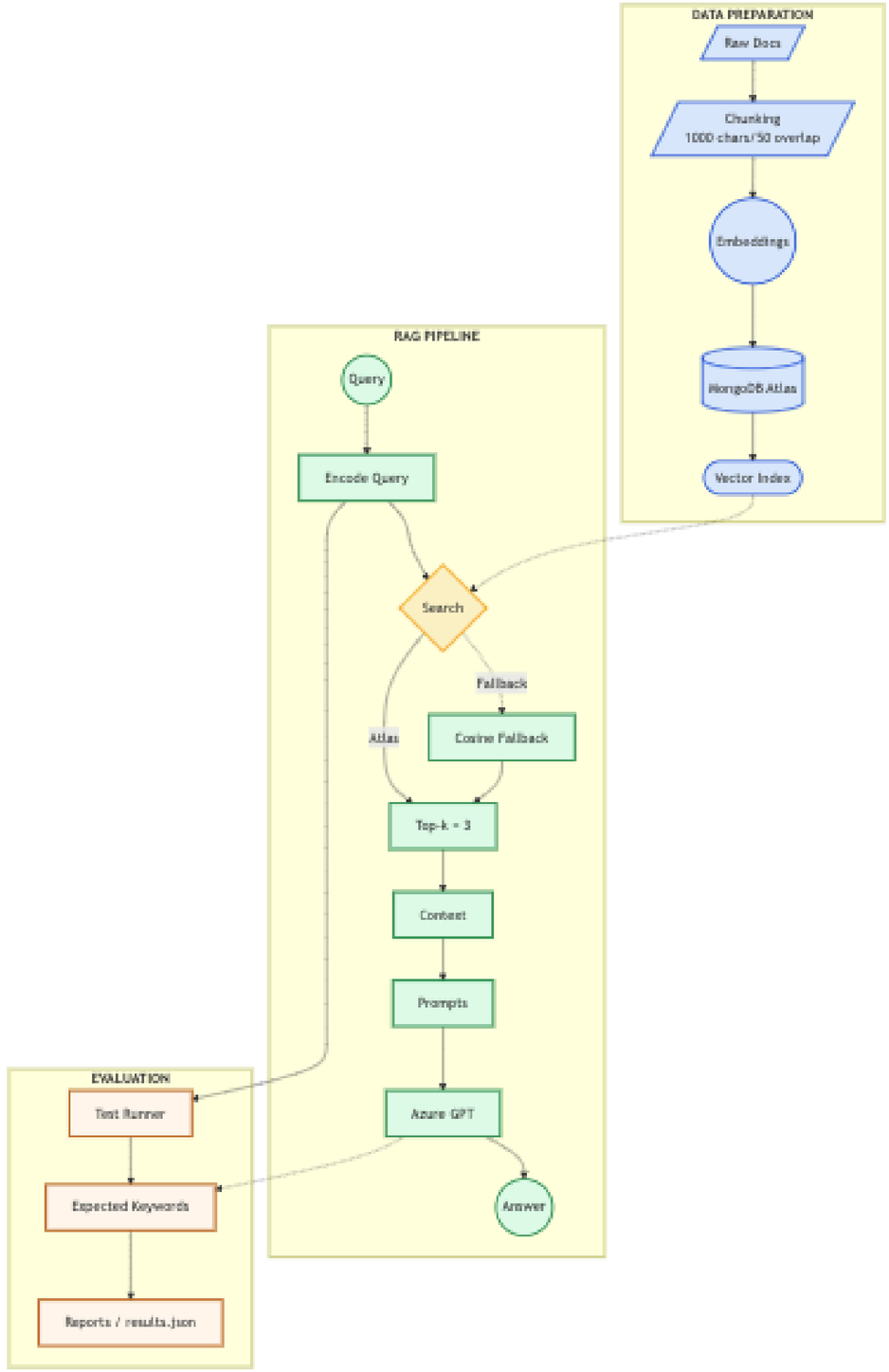
- Raw documents are chunked into manageable pieces
- Embeddings generated using sentence transformers
- Stored in MongoDB Atlas Vector Database

🔄 RAG Pipeline

- User query is encoded into vector format
- System performs vector similarity search
- Top-3 relevant documents retrieved as context
- Context + query sent to Azure GPT-3.5
- AI generates accurate, context-aware answer

💬 Deployment

- Integrated with Discord bot interface
- Real-time query processing
- Seamless user experience



Tools & Technologies

Python + VS Code + Colab

- Backend development environment
- Integration and orchestration
- Colab noetbook - experimentation

Azure OpenAI GPT-3.5 Turbo

- Language model for answer generation
- Context-aware responses

Sentence Transformers

- Model: all-MiniLM-L6-v2 (384 dimensions)
- For document and query embeddings

Discord.py

- Bot framework with token authentication
- Real-time user interaction

MongoDB Atlas

- Vector database for semantic search
- Efficient similarity retrieval

Folder Structure

```
discord-rag-data-scientist/
│
├── backend/                          → Core Python modules
│   ├── chatbot.py                    → Entry point for RAG chatbot logic
│   ├── discord_bot.py                → Connects model to Discord
│   ├── embeddings.py                 → Sentence embeddings (MiniLM / OpenAI)
│   ├── llm.py                        → LLM configuration and responses
│   ├── RAG_pipeline.py               → Retrieval-Augmented Generation pipeline
│   └── retrieval.py                  → Context retrieval logic
│
├── diagrams/                         → System design visuals
│   ├── architectural_diagram.png
│   └── workflow_diagram.png
│
├── docs/                             → Research & documentation
│   ├── EMBEDDING_MODELS_RESEARCH.md
│   ├── IN_SCOPE.md
│   ├── learning_guide.md
│   └── VECTOR_BASES_RESEARCH.md
│
├── notebooks/                        → Model training & evaluation
│   └── Discord_Chatbot_Lab.ipynb
│
├── reports/                          → Evaluation outputs
│   ├── evaluation_report.md
│   └── results.json
│
├── config.py                         → Configuration variables
├── evaluation.py                     → Model evaluation script
├── .env                              → Environment variables (API keys)
├── .gitignore                        → Git ignore rules
├── BLOCKERS.md                       → Issues tracked during development
└── requirements.txt                  → Python dependencies
```

📈 RESULTS & EVALUATION

🎯 SYSTEM PERFORMANCE

✓ 4/4 test queries passed
✓ 69.2% average token overlap
✓ 100% expected keyword match rate

🔍 RETRIEVAL ACCURACY

- Top-3 document retrieval working
- Relevant context successfully extracted
- Source attribution included in responses

💬 RESPONSE QUALITY

- High accuracy: 90-96% token overlap
- Context-aware answer generation
- Graceful handling of out-of-scope queries

(e.g., "Who is Elon Musk?" → "My knowledge base contains information about Python, Machine Learning, Web Development, and Discord.")

⚡ *KEY ACHIEVEMENT*

Fully functional RAG system with robust evaluation framework

Thank You! 🙏

Key Takeaways:

✓ *Built end-to-end RAG system with 100% test accuracy*

✓ *Integrated Azure GPT-3.5 with MongoDB Atlas*

✓ *Deployed functional Discord bot interface*